Available online at www.sciencedirect.com

SciVerse ScienceDirect

Procedia Computer Science 9 (2012) 986 – 993

Procedia Computer Science

International Conference on Computational Science, ICCS 2012

Coupling a Basin Modeling and a Seismic Code using MOAB

Mi Yan[a], Kirk Jordan[b]*, Dinesh Kaushik[c], Michael Perrone[d], Vipin Sachdeva[e], Timothy J. Tautges[f], and John Magerlein[d]

[a]*IBM Systems and Technology Group, 9229 Delegates Row, Suite 500, Indianapolis, Indiana 46240, USA*
[b]*IBM Thomas J. Watson Research Center, 1 Rogers Street, Cambridge MA 02142, USA*
[c]*KAUST Supercomputing Laboratory, King Abdullah University of Science & Technology, Thuwal, Kingdom of Saudi Arabia*
[d]*IBM Thomas J. Watson Research Center, PO Box 218, Yorktown Heights, NY 10598, USA*
[e]*IBM Systems and Technology Group, 1 Rogers Street, Cambridge MA 02142, USA*
[f]*Argonne National Laboratory, Argonne, IL 60439, USA*

Abstract

We report on a demonstration of loose multiphysics coupling between a basin modeling code and a seismic code running on a large parallel machine. Multiphysics coupling, which is one critical capability for a high performance computing (HPC) framework, was implemented using the MOAB open-source mesh and field database. MOAB provides for code coupling by storing mesh data and input and output field data for the coupled analysis codes and interpolating the field values between different meshes used by the coupled codes. We found it straightforward to use MOAB to couple the PBSM basin modeling code and the FWI3D seismic code on an IBM Blue Gene/P system. We describe how the coupling was implemented and present benchmarking results for up to 8 racks of Blue Gene/P with 8192 nodes and MPI processes. The coupling code is fast compared to the analysis codes and it scales well up to at least 8192 nodes, indicating that a mesh and field database is an efficient way to implement loose multiphysics coupling for large parallel machines.

Keywords: Multiphysics coupling; MOAB; basin modeling; seismic

1. Introduction

Computer simulation of large models is undergoing a transition from solving a few heroic problems by a handful of experts to more widespread use in national laboratories, commercial companies, and universities. This transition requires not only domain-specific analysis codes, but also framework software such as that illustrated in Fig. 1 to provide a wide range of capabilities essential for developing and using modern, large-scale simulation tools. These capabilities include communication among codes, meshing, visualization, workflow management, data management, and analytics as well as software development tools such as build systems and quality assurance tools. Here we have focused specifically on multiphysics coupling between different codes used simultaneously to model a problem.

* Corresponding author. Tel.: +1-617-693-4581
E-mail address: kjordan@us.ibm.com

doi:10.1016/j.procs.2012.04.105

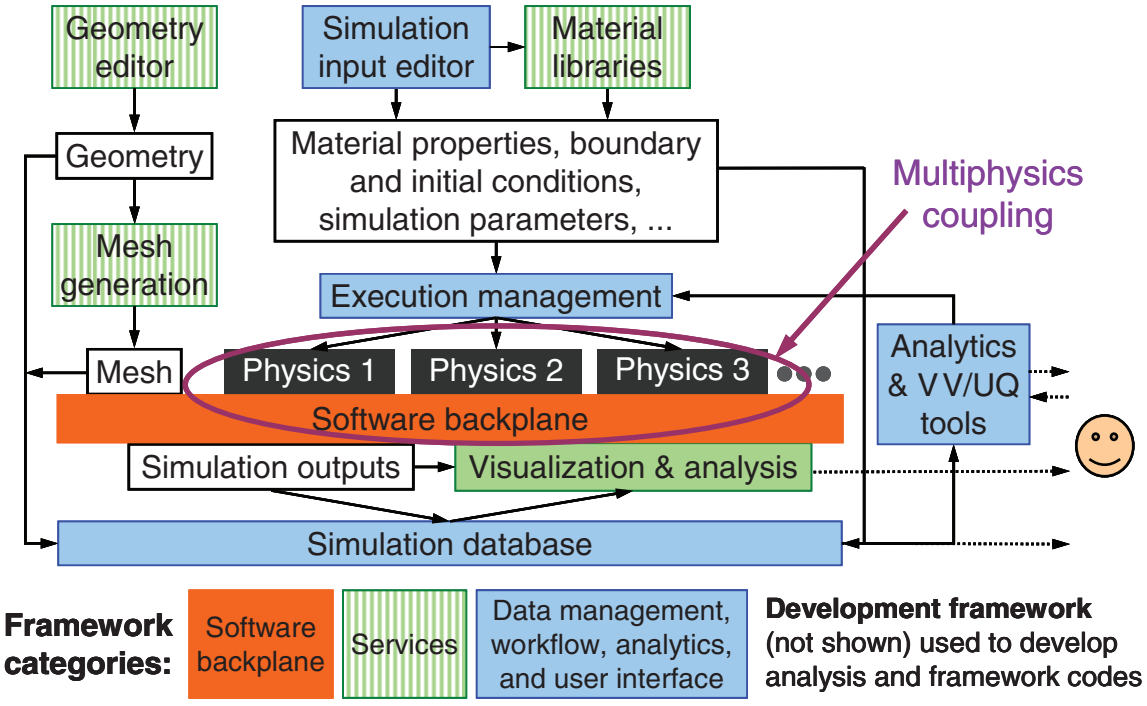

Fig. 1. Example of a set of HPC simulation framework software with colors indicating different categories of software. Unshaded boxes indicate data stored in files and/or memory. Multiphyiscs coupling is provided by a software backplane, which is implemented here as a mesh and field database.

Examples include coupling thermohydraulics with neutronics for nuclear reactor modeling or, as demonstrated here, coupling basin modeling of geological processes with seismic modeling.

One way of implementing multiphysics coupling is through a mesh and field database, which stores mesh and sometimes geometry data as well as input and output field data in memory distributed over the nodes of the parallel machine. Database capabilities include reading mesh data in various formats, partitioning the mesh over the nodes using partitioning codes, handling ghost cells at boundaries between partitions, storing fields from multiple analysis codes, interpolating field data between meshes used by different codes, and storing field data in files at the end of the simulation. All these capabilities must be parallelized with good scaling to handle large problems. Of particular concern is the storage and interpolation of field data since the analysis codes typically use quite different meshes.

Mesh and field databases usually provide for loose coupling, in which the coupled codes run separately for one or more time steps and then exchange data. This approach is easy and minimizes changes to the individual analysis codes, but it may not capture all effects of interest and may sometimes lead to numerical instability. Tighter coupling between codes can be implemented using techniques such as the Jacobian-free Newton-Krylov (JFNK) method, but this is much more intrusive into the analysis codes.

There are presently several mesh and field database implementations of different types [1-6]. These provide generally similar capabilities, but the implementation details differ widely. Despite the existence of these databases, there are relatively few examples in the literature of their use for practical code coupling and we know of no examples in geoscience modeling. Therefore this research has focused on demonstrating and benchmarking coupling between a basin modeling code and a seismic code using a mesh and field database. We have chosen to use the MOAB database [1] from Argonne National Laboratory since it is relatively mature, supports codes written in several languages including C++ and Fortran, is well documented, and is easy to obtain under an attractive LGPL open-source license.

2. Codes Used

2.1. MOAB (Mesh-Oriented database)

MOAB [1] is a component which includes a wide range of capabilities for managing meshes and associated data and for multiphysics code coupling on machines ranging from single processors to large parallel systems. Its libraries, which may be accessed by programs written in C++, C, and Fortran, are designed for parallel performance and memory efficiency. The ITAPS data model used by MOAB includes a full range of geometrical entities, entity sets, and powerful tags used to store fields and other metadata. MOAB supports parallel mesh import and export to/from single HDF5-based files, parallel ghost exchange, communication of field data, and general communication of mesh and metadata among processors. Its MBCOUPLER code provides for linear interpolation of fields between meshes used by different analysis codes, thus providing for multiphysics coupling. Interpolation may be done between meshes occupying roughly the same geometrical space with mesh elements of different types and shapes and with the mesh elements distributed over processes on a parallel machine. MOAB also includes the MBZOLTAN program, which partitions meshes over processes and tags the entities with partition numbers.

Here we have used MOAB to implement practical multiphysics coupling between the PBSM basin modeling code and the FWI3D seismic code.

2.2. PBSM (Parallel BaSin Modeling)

PBSM [7,8] is an experimental parallel basin modeling code written in C++. It is designed to access oil and gas resources in a sedimentary basin by simulating integrated geological processes during the basin evolution including sediment deposition, compaction, tectonic deformation, heat transport, hydrocarbon generation, migration, and accumulation.

PBSM starts by reading an input file describing a mesh for the basin to be modeled. It then uses inputs describing geological events to be simulated and computes temperature and pressure in grid cells at time intervals at the end of each geological event. Here we have modeled only one time interval. Based on the computed temperature and pressure, other field data such as porosity and density are calculated and Gassmann's equations [9] are used to compute an acoustic velocity field at the vertices of the grid. The mesh and the computed velocity field are then ready to be passed to the seismic code.

2.3. FWI3D (Full Wave Inversion in 3 Dimensions)

The FWI3D seismic code [10] implements an iterative refinement approach to 3D seismic imaging using a variant of Full Waveform Inversion (FWI). The code is written in Fortran90. This particular implementation was built on top of KAUST's Reverse Time Migration (RTM) seismic imaging code, which is used as the core engine for generating images. The RTM algorithm is used iteratively to process seismic data into an image that is then used to calculate the discrepancy between the simulations from the current model and the measured seismic data. This discrepancy, in conjunction with a simple line search algorithm, is used as an approximate derivative in model space to reduce the total error and move the current model toward an improved fit to the observed data. The RTM followed by line search is repeated a specified number of time or until some convergence criterion is met.

The RTM algorithm uses a finite difference time domain approach to solve the acoustic wave equation in a 3D volume of the subsurface. Due to stability issues, the algorithm uses a sparse 9x9x9 stencil of 25 elements symmetrically positioned on the axes that is applied over a regular 3D grid. Such a large stencil leads to significant amounts of computation and data movement, which are the primary bottlenecks for this calculation. In order to handle this computational load, the code has been parallelized using MPI to perform domain decomposition, enabling large models to run on thousands of nodes.

3. Implementation of coupling

3.1. MOAB

Our first task was to build the MOAB libraries for the Blue Gene/P system we used here. This presented few problems since MOAB supports the Blue Gene/P platform. We used MOAB release 4.0, the latest publicly available version when we started this work. No changes to MOAB were required to carry out this demonstration. MOAB uses the HDF5 library for file I/O, so this must be installed first.

3.2. Changes to PBSM

Adapting PBSM to use MOAB was relatively easy. We modified PBSM to read from a MOAB mesh file using calls to MOAB rather than its native mesh code and to store the computed acoustic velocity field at the mesh vertices using MOAB. This required changing about 1400 of the 158,000 lines of C++ code in PBSM. No other code changes were required.

Before running PBSM, we partitioned the input file using MBZOLTAN to divide the mesh over the Blue Gene nodes. When partitioning using MBZOLTAN, we normally used the option

```
mbpart --reorder
```

to specify that the mesh file should be written with the vertices belonging to a particular process adjacent in the file. As shown below, this significantly reduces the time for mesh reading.

To estimate relative performance, we ran PBSM using its native mesh input code and also using MOAB mesh management, with the results shown in Fig. 2. For this work, we used the large PBSM model with 48 million tetrahedra described in Section 4 below and the modeling conditions described there. For small numbers of processes, a major fraction of the total time is spent exporting the field data to the MOAB data structures in memory, but this time scales down very well with more nodes. When using native mesh management, the field values were not output, so the times reported are unrealistically short. On the other hand, times for native mesh management include partitioning time not included with MOAB, but this is estimated to be only of order 10 seconds for 1024 processes. Figure 2 also shows that reading an unordered mesh file as described above greatly increases read time, but mesh reading is not a large fraction of the total time. We conclude that using MOAB mesh management is attractive, especially for larger numbers of processes where its excellent scaling is beneficial.

We also compared the amount of memory used on each node by PBSM with MOAB and with the native PBSM

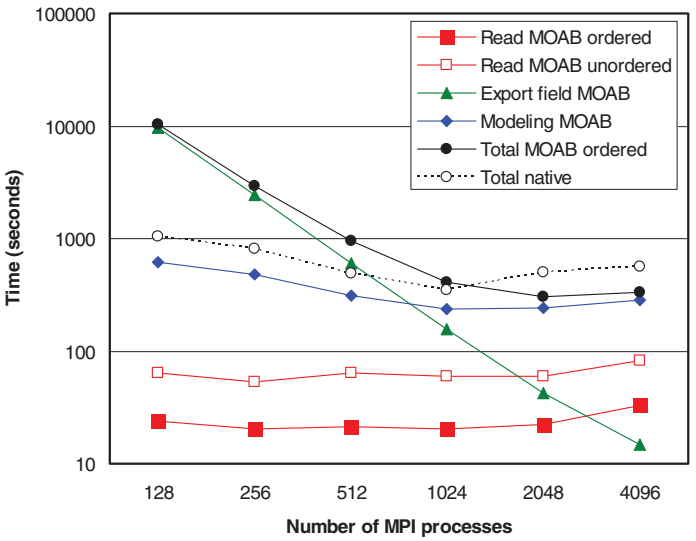

Fig. 2. Execution time for PBSM with MOAB mesh management and native mesh management with the large PBSM model. For a MOAB ordered mesh file, times to read the mesh file, export the field, and carry out the modeling calculation are shown along with the total. The open squares show the time to read an unordered mesh file, while the open circles show the total time with native PBSM mesh management.

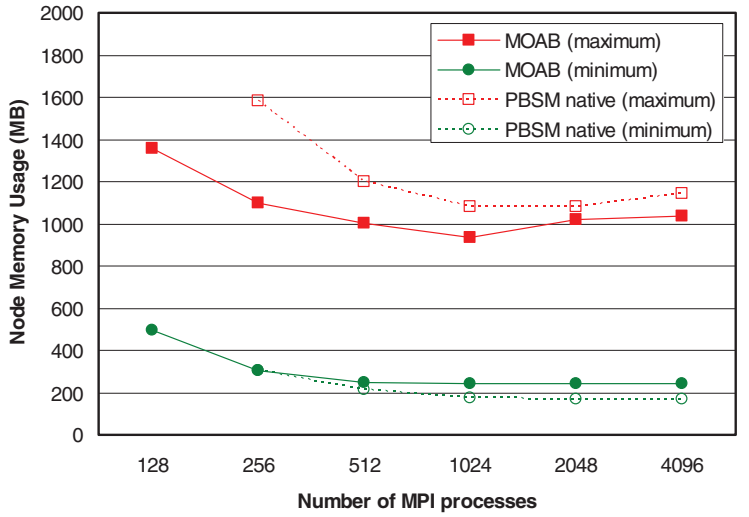

Fig 3. Maximum and minimum memory usage on different nodes by PBSM with MOAB mesh management and native mesh management with the large PBSM model.

mesh management. The memory usage was determined using the Blue Gene kernel function

```
Kernel_GetMemorySize(KERNEL_MEMSIZE_HEAPMAX, &heap);
```

As shown in Fig. 3, varying amounts of memory are used on different nodes of the parallel machine in both cases. This is presumably due to inherent load balancing issues in the original PBSM code. The memory used with MOAB is either comparable to or less than that used with PBSM's native mesh management. While these numbers are not exact due to differences in code functionality as described above, they show that the memory penalty from using MOAB is not large.

3.3. Changes to FWI3D

Adapting FWI3D to use MOAB was similarly easy. It was modified to read the acoustic velocity field through calls to MOAB. This required changing about 250 of the 7000 lines of Fortran90 code. We did not enable FWI3D write its output using MOAB, but this would be straightforward.

3.4. Performing coupled simulations

To carry out coupled simulations, we created a single executable consisting of PBSM, the MOAB mesh-mesh coupler MBCOUPLER, and FWI3D. This requires that each MPI process have enough memory for all three codes and their data. Therefore some care must be taken with memory management in the codes to be coupled. Memory for data can generally be managed by using more nodes for solving larger problems, but large global arrays and code size must be minimized. Here we found that the HDF5 collective file I/O operations required a large amount of buffer space on each node and sometimes overflowed the available memory. Therefore we sometimes found it necessary to direct HDF5 to use point-to-point collective file I/O operations (`MPI_Isend/Irecv`) rather than using the default `MPI_Alltoallv`. This is done by setting the environment variable `BGLMPIO_COMM=1`. This change slowed reading of mesh data files into MOAB by roughly a factor of 2, but the read time is still a small fraction of the total and is thus quite acceptable.

To carry out a coupled simulation, PBSM uses its inputs and, as described above, computes an acoustic velocity field at the vertices of its grid. The mesh and the velocity field data generated by PBSM are stored in memory using MOAB. Next, the MBCOUPLER part of MOAB uses linear interpolation to convert the velocity field from the PBSM mesh to the different mesh used by the FWI3D seismic code. Finally, FWI3D obtains the velocity field on its mesh from MOAB and carries out a seismic calculation as described above.

4. Benchmarking results

All benchmarking was carried using one of two models, a small and a large model described in Table 1. The table gives the sizes of the PBSM and FWI3D meshes for these two models. PBSM uses an unstructured mesh, but here we used a mesh with vertices in a regular rectangular array. PBSM uses tetrahedral mesh cells to carry out its calculation and calculates the acoustic velocity field at the mesh vertices. The seismic code internally uses a structured mesh consisting only of vertices, but this is put into MOAB as an unstructured mesh. MOAB carries out interpolation between two unstructured meshes to calculate the acoustic velocity at each of the FWI3D vertices as required by the program. We note that the partitioning of the two meshes is completely independent so that a particular region of space in one mesh is likely to be handled by a different MPI process than that region in the other mesh.

Table 1. Size of the meshes for PBSM and FWI3D for the small and large models used here.

	Small Model	Large Model
PBSM: Number of vertices in x,y,z	101x101x101	201x201x201
PBSM: Vertex spacing in x,y,z (m)	40, 40, 20	20, 40, 10
PBSM: Number of tetrahedra in grid	About 6 million	About 48 million
FWI3D: Number of vertices in x,y,z	384x384x192	384x384x192
FWI3D: Vertex spacing in x,y,z (m)	10, 10, 10	10, 10, 10

Benchmarking was carried out on the Shaheen Blue Gene/P system at KAUST. The system consists of 16 racks, each with 1024 compute nodes. Benchmarking was done on racks with 128 compute nodes for each I/O node. One MPI process was run on each compute node so that the full 4 GB of memory on the node was available to that process. We specified HDF5 point-to-point collective file I/O in all cases (see section 3.4). The execution time was measured for PBSM, the MBCOUPLER coupling code which does mesh-mesh interpolation, and FWI3D. The results are plotted in Fig. 4.

We see that PBSM scales well up to 1024 processes for the large model. For more processes and for the small model, there appears to be insufficient work to offset the larger communication and other overhead from adding more processes. FWI3D scales relatively well all the way to 8192 processes.

The time to interpolate between the two meshes is small compared to the time used by the analysis codes and it scales very well all the way to at least 8192 processes. This result indicates that using the MOAB mesh and field database is an efficient way to implement loose multiphysics coupling for large parallel machines.

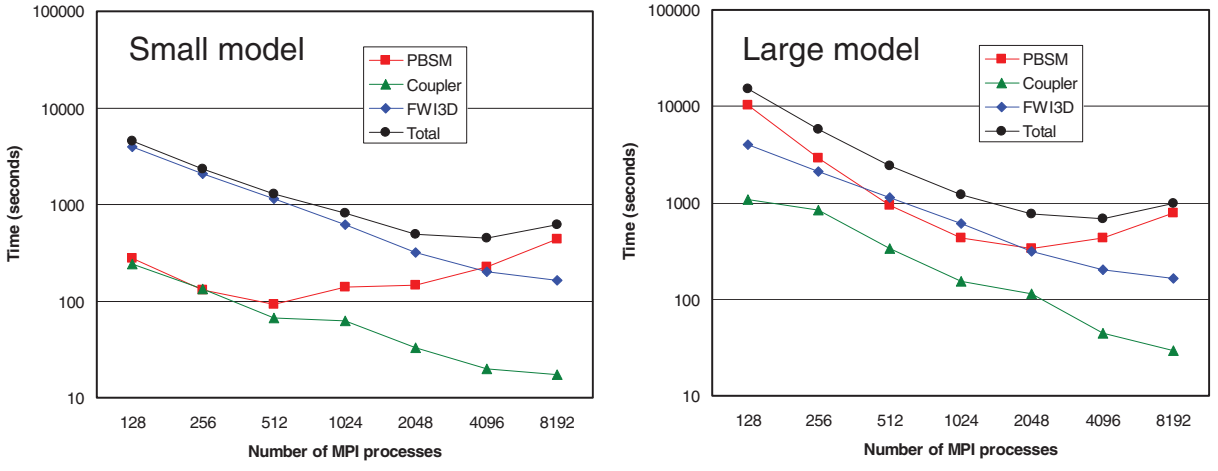

Fig. 4. Execution time for the different parts of the small model (left) and large model (right).

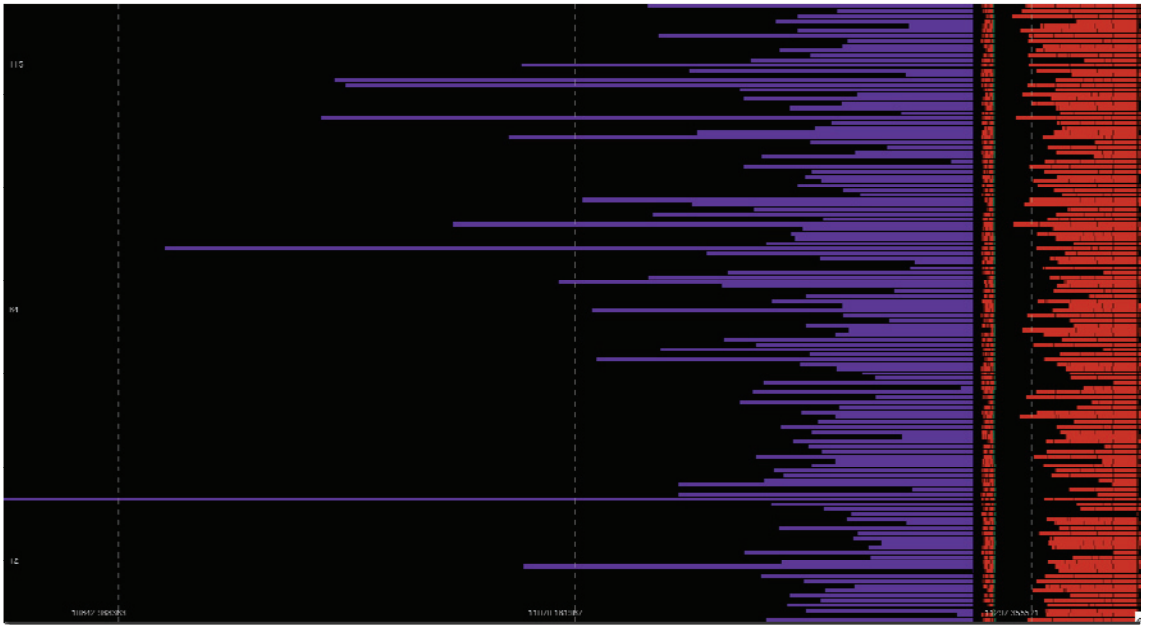

Fig. 5. Output from mpitrace for the large model running on 128 nodes showing the MPI utilization for MBCOUPLER for processes 0 to 127 from bottom to top with time going from left to right. The black segments indicate computation by a process, the purple in the middle indicates waiting at `MPI_Allgather`, and the orange at the right waiting at `MPI_Waitall`.

In order to understand the efficiency of the MOAB coupling code in more detail, we used an MPI profiler for the large model running on 128 nodes to show what each process is doing during the time MBCOUPLER is running. The results are shown in Fig. 5. We see that most processes are doing computation more than 75% of the time, with no processes having grossly too much work to do and only a few processes having significantly too little. This good load balancing is consistent with the scaling behavior observed for MBCOUPLER.

We also examined the memory used by the various MPI processes when running the fully coupled code using the large model. Figure 6 plots the minimum memory used by any process, the average memory used, and maximum memory used. The memory use is dominated by PBSM and the maximum memory is close to that used by PBSM

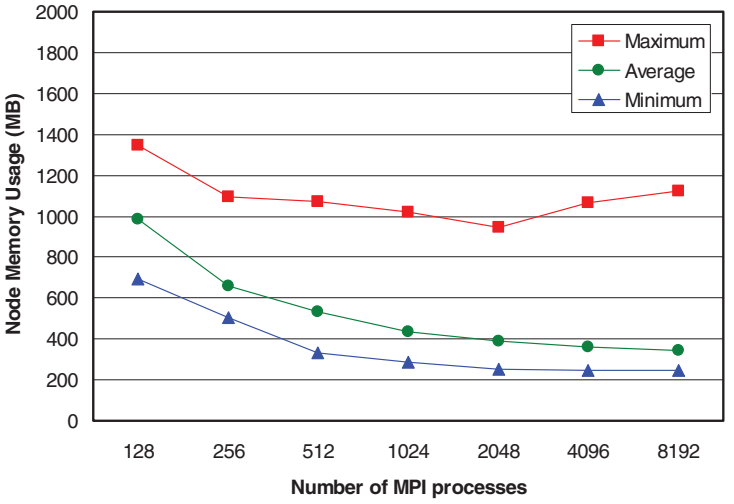

Fig. 6. Minimum, average, and maximum memory usage by the different MPI processes for the large model.

alone. The average memory tracks the minimum memory usage since only a small number of processes use an unusually large amount of memory when PBSM is running. Further investigation of PBSM memory use would be required to minimize overall memory usage. For most processes, memory use is reduced as the data are spread across more processes.

5. Conclusions

We have successfully used the MOAB open-source mesh and field database to implement multiphysics coupling between a basin modeling code and a seismic code. The codes to be coupled could be modified to use MOAB without major disruption or performance degradation. MOAB showed excellent performance and scaling both for storage of field data and for interpolation of fields between the different meshes used by the coupled codes. Now that we have demonstrated that multiphysics coupling using MOAB is practical for these experimental codes, we anticipate addressing a problem of real scientific interest requiring coupling of two or more production codes.

Acknowledgements

We gratefully acknowledge valuable consultations with Sanzong Zhang from G. Schuster's team at KAUST and Karen Magerlein from IBM Research on the use of FWI3D. In addition, we would like to thank Mary F. Wheeler and Carlos Torres-Verdin from the University of Texas at Austin for helpful conversations on the physics of coupling a seismic code with a basin model that represents the reservoir.

References

1. MOAB, http://trac.mcs.anl.gov/projects/ITAPS/wiki/MOAB.
2. STKmesh, http://trilinos.sandia.gov/packages/stk.
3. Salome: The Open Source Platform for Numerical Simulation, http://www.salome platform.org.
4. Flexible Mesh DataBase (FMDB), http://www.scorec.rpi.edu/FMDB.
5. libmesh, http://libmesh.sourceforge.net.
6. Rocstar, http://www.csar.illinois.edu/rocstar.
7. U. T. Mello, J. R. Rodrigues, and A, Rossa, A Control-Volume Finite-Element Method for Three-Dimensional Multiphase Basin Modeling, Marine and Petroleum Geology, 26:504-518 (2009).
8. U. T. Mello and I. Khabibrakhmanov, An Efficient Mesh Management for Moving Boundaries Associated with Erosion Events, AAPG Hedberg Conference, Basin and Petroleum System Modeling: New Horizons in Research and Applications, Napa, CA, May 3-7, 2009.
9. Gary Mavko, Tapan Mukerji, and Jack Dvorkin, The Rock Physics Handbook: Tools for Seismic Analysis of Porous Media, second edition, Cambridge University Press, 2009.
10. Chaiwoot Boonyasiriwat, and Gerard Schuster, 3D Multisource Full-Waveform Inversion using Dynamic Quasi-Monte Carlo Phase Encoding, Geophysical Research Abstracts Vol. 12, 2010 EGU General Assembly 2010.